Sequência Projeto - Multi Cloud, Multi Linguagem, Serverless, Microsserviços

OBS: Todos os códigos estão no github:
https://github.com/danferreira011/Multicloud-Microservicos.git

Parte 1: AWS Lambda + API Gateway (Python)

Esta função converte Celsius para Fahrenheit (°C -> °F)

1. **Criar a Função Lambda:**
   - Acesse o Console AWS -> Lambda
   - Clique em "Create function"
   - Selecione "Author from scratch"
   - Nome da Função: converteCelsiusParaFahrenheit
   - Runtime: Python 3.12 (ou mais recente)
   - Arquitetura: x86_64
   - Permissões: "Create a new role with basic Lambda permissions"
   - Clique em "Create function"
2. **Adicionar o Código Python:**

   Na aba "Code", cole o seguinte no lambda_function.py:

```
Python


import json

def lambda_handler(event, context): # Assinatura corrigida
    try:
        # API Gateway envia 'body' como string
        if 'body' not in event or event['body'] is None:
            raise ValueError("Corpo da requisição está vazio ou ausente.") # Mensagem corrigida

        body = json.loads(event['body'])
        celsius_str = body.get('celsius')
```

```
        if celsius_str is None:
            raise ValueError("Chave 'celsius' não encontrada no corpo da requisição.")
    # Mensagem corrigida

        celsius = float(celsius_str) # Conversão corrigida
        fahrenheit = (celsius * 9/5) + 32 # Fórmula corrigida

        response_body = {
            "fahrenheit_calculado": fahrenheit, # Chave corrigida
            "celsius_recebido": celsius # Chave corrigida
        }

        return {
            "statusCode": 200,
            "headers": {
                "Access-Control-Allow-Origin": "*", # Aspas corrigidas
                "Access-Control-Allow-Headers": "Content-Type",
                "Access-Control-Allow-Methods": "POST, OPTIONS" # GET removido
            },
            "body": json.dumps(response_body) # Variável corrigida
        }

    except Exception as e:
        return {
            "statusCode": 400,
            "headers": {
                "Access-Control-Allow-Origin": "*", # Aspas corrigidas
                "Access-Control-Allow-Headers": "Content-Type",
                "Access-Control-Allow-Methods": "POST, OPTIONS" # GET removido
            },
            # Erro de digitação corrigido: faltava chave "erro"
            "body": json.dumps({"erro": str(e)})
        }
```

Clique em **"Deploy"**.

3. **Testar a Lambda (Opcional, mas recomendado):**
   - Vá para a aba "Test".
   - Crie um evento de teste com o nome testeCelsius.
   - No "Event JSON", cole:

```
JSON
{
  "body": "{\"celsius\": 20}"
}
```

- Clique em "Save" e depois em "Test". Verifique se o resultado é sucesso (200 OK).

4. **Criar o Gatilho REST API:**
   - Na página da Lambda, clique em "+ Add trigger".
   - Selecione API Gateway.
   - Escolha "Create a new API".
   - Tipo de API: **REST API**.
   - Segurança: **Open**.
   - Clique em "Add".
5. **Configurar CORS na REST API:**
   - Aguarde o gatilho ser criado.
   - Vá para o serviço **API Gateway**.
   - Clique na sua nova REST API (ex: converteCelsiusParaFahrenheit-API) .
   - Em "Resources", clique no recurso da sua função (ex: /converteCelsiusParaFahrenheit) .
   - Clique no menu "Actions" -> "Enable CORS".
   - Aceite os padrões (POST, OPTIONS, Origin '*') e clique em "Enable CORS and replace...".
6. **Implantar (Deploy) a REST API:**
   - Clique no menu "Actions" -> "Deploy API" .
   - Deployment stage: "[New Stage]".
   - Nome do Estágio: prod (ou v1).

- Clique em "Deploy".

7. **Obter a URL da AWS:**
   - Após o deploy, copie a "Invoke URL" fornecida (ex: https://.../prod/converteCelsiusParaFahrenheit) .
   - Guarde esta URL para o index.html.

Parte 2: Azure Functions (PowerShell/Windows)

Esta função converte Fahrenheit para Celsius (°F -> °C).

1. **Criar o Function App:**
   - Portal Azure -> "+ Create a resource" -> "Function App" .
   - Grupo de Recursos: Novo (ex: rg-portfolio-windows).
   - Nome do App: Único (ex: app-conversor-pshell).
   - Publicar: Code.
   - Runtime stack: **PowerShell Core** (ex: 7.2 ou 7.4) .
   - Sistema Operacional: **Windows** (será selecionado automaticamente) .
   - Plano: **Consumption (Serverless)**.
   - Clique em "Review + create" -> "Create".
2. **Criar a Função:**
   - Aguarde a implantação e vá para o Function App.
   - Clique em "Functions" -> "+ Create".
   - Gatilho: HTTP trigger.
   - Nome da Função: converteFahrenheitParaCelsius.
   - Nível de autorização: **Anonymous**.
   - Clique em "Create".
3. **Adicionar o Código PowerShell:**

- Clique na função criada.
- Vá para "Code + Test".
- No run.ps1, cole:

PowerShell

```
param($Request, $TriggerMetadata)

# 1. O Azure já "converteu" o JSON para nós porque viu o "Content-Type".
# O $Request.Body JÁ É um objeto PowerShell.
$data = $Request.Body

# 2. Pega o valor 'fahrenheit' diretamente do objeto
$fahrenheit_str = $data.fahrenheit
if (-not $fahrenheit_str) {
    # Se a chave 'fahrenheit' não for encontrada...
    Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
        StatusCode = 400
        Body = "Chave 'fahrenheit' não encontrada no corpo da requisição."
    })
    return
}

# 3. A fórmula de conversão
try {
    $fahrenheit = [float]$fahrenheit_str
    $celsius = ($fahrenheit - 32) * 5/9
}
catch {
    # Se 'fahrenheit' não for um número (ex: "abc")
    Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
        StatusCode = 400
        Body = "Valor 'fahrenheit' não é um número válido."
    })
    return
}


# 4. Prepara o corpo da resposta
```

```
$responseBody = @{
    celsius_calculado = $celsius
    fahrenheit_recebido = $fahrenheit

} | ConvertTo-Json

# 5. Retorna a resposta de sucesso (200)
Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
    StatusCode = 200
    Body = $responseBody
    Headers = @{
        'Content-Type' = 'application/json'
    }
})
```

- Clique em "Save".

4. **Configurar CORS (Métodos e Cabeçalhos):**
   - No menu da função, clique em "Integration".
   - Clique em "HTTP (req)".
   - Marque a caixa **OPTIONS** em "Allowed HTTP Methods" .
   - Clique em "Save".
   - Volte para a página principal do Function App.
   - Vá em "CORS" (seção API).
   - Adicione * em "Allowed Origins".
   - Adicione Content-Type em "Allowed Headers".
   - Certifique-se que POST e OPTIONS estão em "Allowed Methods".
   - Clique em "Save".

5. **Obter a URL do Azure:**
   - Volte para a função converteFahrenheitParaCelsius.
   - Clique em "Get Function Url".
   - Copie a URL.

- Guarde esta URL para o index.html.

Parte 3: GCP Cloud Run (PHP)

Esta função converte Celsius para Kelvin (°C -> °K).

1. **Habilitar APIs (se necessário):**
   - No Google Cloud Console, ative "Cloud Functions API" e "Cloud Build API".
2. **Criar o Serviço Cloud Run:**
   - Vá para Cloud Run -> "Create Service".
   - Selecione "Deploy one revision..." -> **"Use an inline editor to create a function"**.
   - Service name: convertecelsiusparakelvin.
   - Region: Sua preferência (ex: europe-west1).
   - Runtime: **PHP 8.3** (ou mais recente).
   - Authentication: **"Allow unauthenticated invocations"**.
   - Clique em "Create".
3. **Adicionar o Código PHP:**
   - Aguarde a criação e o editor inline aparecer.
   - No arquivo index.php, cole :

```
PHP

<?php

// --- MANIPULAÇÃO DE CORS ---
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: POST, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type");

// 1. Responde ao "preflight" (requisição OPTIONS)
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
```

```
    http_response_code(204); // No Content
    exit;
}
// ——————————————————————————

// 2. Garante que só aceitamos POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405); // Method Not Allowed
    echo json_encode(['erro' => 'Método não permitido. Use POST.']);
    exit;
}

// 3. Pega o corpo JSON da requisição
$json_data = file_get_contents('php://input');
$data = json_decode($json_data, true); // O 'true' converte para array associativo

if ($data === null || !isset($data['celsius'])) {
    http_response_code(400); // Bad Request
    echo json_encode(['erro' => "Chave 'celsius' não encontrada ou JSON inválido no corpo da requisição."]);
    exit;
}

$celsius_str = $data['celsius'];

// 4. Valida e converte para float
if (!is_numeric($celsius_str)) {
    http_response_code(400); // Bad Request
    echo json_encode(['erro' => "Valor 'celsius' não é um número válido."]);
```

```
    exit;
}

$celsius = (float)$celsius_str;

// 5. A fórmula de conversão
$kelvin = $celsius + 273.15;

// 6. Prepara a resposta de sucesso
$responseBody = [
    'kelvin_calculado' ⇒ $kelvin,
    'celsius_recebido' ⇒ $celsius,
];

// 7. Retorna a resposta
http_response_code(200); // OK
header('Content-Type: application/json');
echo json_encode($responseBody);

?>
```

(PHP não precisa de outros arquivos de configuração aqui).

4. **Deploy:**
   - Clique em "Deploy".

5. **Obter a URL do GCP:**
   - Após o deploy, copie a **URL** do serviço Cloud Run (ex: https://convertecelsiusparakelvin-...a.run.app) .

   - Guarde esta URL para o index.html.

Parte 4: HTML Final e Teste Integrado

1. **Código index.html:**
   - Crie um arquivo index.html com o código fornecido .

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Conversor Multicloud (AWS + Azure + GCP)</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            min-height: 90vh;
            background-color: #f0f2f5;
            gap: 1.5rem;
            margin-top: 2rem;
        }
        .container {
            background: #fff;
            padding: 2rem;
            border-radius: 8px;
            box-shadow: 0 4px 12px rgba(0,0,0,0.1);
            text-align: center;
            width: 450px;
        }
        .container h1 {
            margin-top: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            gap: 10px;
            font-size: 1.8rem;
        }
        .container img {
            height: 35px;
        }
        input {
            font-size: 1.2rem;
            padding: 0.5rem;
            margin: 1rem 0;
            width: 150px;
```

```
            text-align: center;
        }
        button {
            font-size: 1rem;
            padding: 0.6rem 1rem;
            cursor: pointer;
            color: white;
            border: none;
            border-radius: 4px;
            margin: 0 5px;
            transition: transform 0.1s ease;
        }
        button:active {
            transform: scale(0.98);
        }
        .aws { background-color: #FF9900; }
        .aws:hover { background-color: #e68a00; }

        .azure { background-color: #0078D4; }
        .azure:hover { background-color: #005a9e; }

        .gcp { background-color: #4285F4; }
        .gcp:hover { background-color: #357ae8; }

        .resultado {
            margin-top: 1.5rem;
            font-size: 1.5rem;
            font-weight: bold;
            color: #333;
            min-height: 2.2rem; /* Evita que a caixa "pule" */
        }
    </style>
</head>
<body>

    <div class="container">
        <h1>
            Conversor Celsius (°C)
        </h1>
        <p>Digite a temperatura em Celsius</p>

        <div>
            <input type="number" id="celsiusInput"
```

```
placeholder="Ex: 20">
        </div>
        <div>
            <button id="convertButtonAws" class="aws">
                <img
src="https://upload.wikimedia.org/wikipedia/commons/9/93/Amazon_
Web_Services_Logo.svg" style="height:15px; vertical-align:
middle; margin-right: 5px;">
                para °F (AWS)
            </button>
            <button id="convertButtonGcp" class="gcp">
                <img
src="https://upload.wikimedia.org/wikipedia/commons/2/2f/Google_
Cloud_logo.svg" style="height:15px; vertical-align: middle;
margin-right: 5px;">
                para °K (GCP)
            </button>
        </div>

        <div id="resultadoAws" class="resultado">
            </div>
        <div id="resultadoGcp" class="resultado">
            </div>
    </div>

    <div class="container">
        <h1>
            <img
src="https://upload.wikimedia.org/wikipedia/commons/a/a8/Microso
ft_Azure_Logo.svg" alt="Azure Logo">
            Conversor (°F)
        </h1>
        <p>Digite a temperatura em Fahrenheit</p>

        <div>
            <input type="number" id="fahrenheitInput"
placeholder="Ex: 68">
        </div>
        <div>
            <button id="convertButtonAzure" class="azure">
                para °C (Azure)
            </button>
        </div>
```

```
        <div id="resultadoAzure" class="resultado">
            </div>
    </div>


    <script>
        // --- CONFIGURAÇÃO DAS URLs ---

        // 1. Cole a URL da sua API AWS (REST API)
        // Ex:
"https:// ... execute-api.us-east-1.amazonaws.com/prod/converteCel
siusParaFahrenheit"
        const awsApiUrl = "COLE_SUA_URL_DA_AWS_AQUI";

        // 2. Cole a URL da sua Azure Function (PowerShell ou
PHP)
        // Ex:
"https://app-conversor-pshell ... azurewebsites.net/api/converteFa
hrenheitParaCelsius"
        const azureApiUrl = "COLE_SUA_URL_DA_AZURE_AQUI";

        // 3. Cole a URL da sua GCP Cloud Run (PHP ou Go)
        // Ex: "https://convertecelsiusparakelvin- ... a.run.app"
        const gcpApiUrl = "COLE_SUA_URL_DA_GCP_AQUI";


        // --- LÓGICA DO BLOCO 1 (CELSIUS) ---
        const celsiusInput =
document.getElementById("celsiusInput");

        // Botão AWS
        const convertButtonAws =
document.getElementById("convertButtonAws");
        const resultadoAwsDiv =
document.getElementById("resultadoAws");

        convertButtonAws.addEventListener("click", () ⇒ {
            const celsiusValue = celsiusInput.value;
            if (celsiusValue ≡≡ "") {
                resultadoAwsDiv.textContent = "Digite um valor
Celsius.";
                resultadoGcpDiv.textContent = "";
```

```
                return;
            }
            resultadoAwsDiv.textContent = "Convertendo
(AWS)...";
            resultadoGcpDiv.textContent = ""; // Limpa o outro
resultado

            fetch(awsApiUrl, {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ celsius: celsiusValue })
            })
            .then(response ⇒ {
                if (!response.ok) { // Verifica se a resposta
não foi OK (ex: 4xx, 5xx)
                    throw new Error(`Erro HTTP AWS:
${response.status}`);
                }
                return response.json();
            })
            .then(data ⇒ {
                if (data.fahrenheit_calculado ≠= undefined) {
                    let fahrenheit =
data.fahrenheit_calculado.toFixed(2);
                    resultadoAwsDiv.innerHTML =
`${celsiusValue}°C é igual a <strong>${fahrenheit}°F</strong>`;
                } else {
                    resultadoAwsDiv.textContent = "Erro AWS: " +
(data.erro || data.message || "Resposta inválida");
                }
            })
            .catch(error ⇒ {
                console.error("Erro ao chamar a API AWS:",
error);
                resultadoAwsDiv.textContent = `Erro API AWS:
${error.message}`;
            });
        });

        // Botão GCP
        const convertButtonGcp =
document.getElementById("convertButtonGcp");
        const resultadoGcpDiv =
```

```
document.getElementById("resultadoGcp");

        convertButtonGcp.addEventListener("click", () => {
            const celsiusValue = celsiusInput.value;
            if (celsiusValue === "") {
                resultadoGcpDiv.textContent = "Digite um valor Celsius.";
                resultadoAwsDiv.textContent = "";
                return;
            }
            resultadoGcpDiv.textContent = "Convertendo (GCP)...";
            resultadoAwsDiv.textContent = ""; // Limpa o outro resultado

            fetch(gcpApiUrl, {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ celsius: celsiusValue })
            })
            .then(response => {
                 if (!response.ok) {
                    throw new Error(`Erro HTTP GCP: ${response.status}`);
                }
                return response.json();
            })
            .then(data => {
                if (data.kelvin_calculado !== undefined) {
                    let kelvin = data.kelvin_calculado.toFixed(2);
                    resultadoGcpDiv.innerHTML = `${celsiusValue}°C é igual a <strong>${kelvin}°K</strong>`;
                } else {
                    resultadoGcpDiv.textContent = "Erro GCP: " + (data.erro || "Resposta inválida");
                }
            })
            .catch(error => {
                console.error("Erro ao chamar a API GCP:", error);
                resultadoGcpDiv.textContent = `Erro API GCP: ${error.message}`;
```

```
            });
        });


        // --- LÓGICA DO BLOCO 2 (FAHRENHEIT → AZURE) ---
        const fahrenheitInput =
document.getElementById("fahrenheitInput");
        const convertButtonAzure =
document.getElementById("convertButtonAzure");
        const resultadoAzureDiv =
document.getElementById("resultadoAzure");

        convertButtonAzure.addEventListener("click", () => {
            const fahrenheitValue = fahrenheitInput.value;
            if (fahrenheitValue === "") {
                resultadoAzureDiv.textContent = "Digite um valor
Fahrenheit.";
                return;
            }
            resultadoAzureDiv.textContent = "Convertendo
(Azure)...";

            fetch(azureApiUrl, {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ fahrenheit:
fahrenheitValue })
            })
            .then(response => {
                 if (!response.ok) {
                    throw new Error(`Erro HTTP Azure:
${response.status}`);
                }
                return response.json();
            })
            .then(data => {
                if (data.celsius_calculado !== undefined) {
                    let celsius =
data.celsius_calculado.toFixed(2);
                    resultadoAzureDiv.innerHTML =
`${fahrenheitValue}°F é igual a <strong>${celsius}°C</strong>`;
                } else {
                    resultadoAzureDiv.textContent = "Erro Azure:
```

```
    " + (data.erro || "Resposta inválida");
                }
            })
            .catch(error => {
                console.error("Erro ao chamar a API Azure:", error);
                resultadoAzureDiv.textContent = `Erro API Azure: ${error.message}`;
            });
        });

    </script>

</body>
</html>
```

- **IMPORTANTE:** Substitua os placeholders COLE_SUA_URL_DA_AWS_AQUI, COLE_SUA_URL_DA_AZURE_AQUI e COLE_SUA_URL_DA_GCP_AQUI pelas URLs reais que você copiou.

2. **Teste Final:**
   - Salve o index.html.
   - Rode um servidor web local na pasta do arquivo (ex: python -m http.server 8080 ou use o Live Server do VS Code).
   - Abra a página no navegador (ex: http://localhost:8080).
   - Faça um Hard Refresh (Ctrl + Shift + R).
   - Teste as três conversões:
     - Celsius para Fahrenheit (AWS).
     - Celsius para Kelvin (GCP).
     - Fahrenheit para Celsius (Azure).